“Barcode” web service
Instructions for access in PHP

Version 2.2
November 2013

SWISS POST

Table of contents

References

[1] "Barcode" web service, manual, http://www.swisspost.ch/post-barcode-handbuch.pdf

[2] Documentation about "Barcode" web service, http://www.swisspost.ch/post-barcode-cug.htm

1 Introduction

This documentation explains how to use the "Barcode" web service in PHP. The use of the web service is explained by means of a number of typical examples.

This documentation is not intended to provide a complete specification of the web service interface, but simply contains instructions on how the web service can be used in PHP.

For full documentation of the "Barcode" web service, please refer to the Web Service User Manual [1], which also explains all the terms, attributes, service codes etc. Additional resources and tools for the "Barcode" web service can be found on the "Barcode" web service homepage [2].

All examples are also available as php sources. The examples illustrated here use a couple of simple aids that are available in the **wsbc-utils.php** file.

2 Configuration

2.1 WSDL files

The SOAPClient php has a problem with accessing WSDL files via proxies. We therefore strongly advise that you save the WSDL file and associated XSD file locally for access from php.

To do this, download the following two XML files and save them under the file names provided in a directory that can be accessed in php. For the following examples, we have assumed that these files are in the same directory as the php sources.

For the following examples it is assumed that these files are located in the same directory as the PHP source files.

URL for download	Save under file name
http://www.post.ch/post-barcode-description-service.wsdl	barcode_v2_2.wsdl
http://www.post.ch/barcode_v2_2.xsd	barcode_v2_2.xsd

ATTENTION: Both files must be located in the same directory, and the file barcode_v2_2.xsd must bear this exact name, since this file is integrated in the WSDL file.

These files are also contained in the php example sources.

2.2 SOAPClient configuration

The web service is accessed via a SOAPClient instance. The following example illustrates the initialization of the SOAPClient object for accessing the "Barcode" web service.

The parameters that have been commented out are optional and can be used if needed.

```
// SOAP Configuration

$username = 'your username';
$password = 'your password';
$endpoint_url= 'http://anylocationURL/v2_2';
$SOAP_wsdl_file_path= 'barcode_v2_2.wsdl';

$SOAP_config = array(
    // Webservice Endpoint URL
    'location' => 'https://wsbc.post.ch/wsbc/barcode/v2_2,

    // Webservice Barcode Login
    'login' => $username,
    'password' => $password,

    // Encoding for Strings
    // 'encoding' => 'ISO-8859-1',

    // Optional Proxy Config
    // (if you are behind a proxy):
    // 'proxy_host' => 'proxy.mydomain.com',
    // 'proxy_port' => 8080,

    // Optional Proxy Authentication
    // (if your proxy needs a username and password):
    // 'proxy_login' => 'proxy-username',
    // 'proxy_password' => 'proxy-password',

    // Addtional debug trace information:
    // 'trace' => true,

    // Connection timeout (in seconds):
    // 'connection_timeout' => 90

);

// SOAP Client Initialization
try {
    $SOAP_Client = new SoapClient($SOAP_wsdl_file_path, $SOAP_config);
}
catch (SoapFault $fault) {
    echo('Error in SOAP Initialization: '. $fault -> __toString() .'<br/>');
    exit;
}
```

A sample configuration can also be found in the **wsbc-init.php** file that is incorporated in all the other examples.

In order for the examples to work, a valid username and password for access to the "Barcode" web service must be entered in the configuration.

All the examples below assume that a corresponding SOAPClient has first been correctly initialized.

2.3 Proxy configuration

Any proxy for access to the Internet must be properly configured in the **SOAP_config** array. See the commented out lines in the previous example in section 2.2.

2.4 Encoding character set

The SOAPClient can be configured to work with character set encoding other than UTF-8. To do this, see the "encoding" setting in the previous example in section 2.2. This setting means that all the strings passed in the encoding set are expected and duly converted to UTF-8. All return values are also converted back by UTF-8to the character set duly set.

In this setting therefore, you should, if possible, set the encoding that is used in the php sources to avoid the strings having to be constantly encoded.

The php sample sources that are also supplied are coded in UTF-8, which is why this setting is not included in the examples.

3 Read operations

The read operations can be used to call up all the service codes and label layouts of the web service that are currently supported.

The following php example illustrates how all these read operations are used. The example illustrates a handful of all the service codes and label layouts currently available in HTML.

```
include("wsbc-init.php");
include("wsbc-utils.php");

// covered requests:
// 1. ReadServiceGroups
// 2.1 ReadBasicServices
// 2.1.1 ReadAdditionalServices
// 2.1.2 ReadDeliveryInstructions
// 2.1.3 ReadLabelLayouts
// 3. ReadAllowedServicesByFrankingLicense

echo "<html><header></header><body>";

//
// 1. Read all service groups
$serviceGroupsRequest = array('Language' => 'de');
try {
    $serviceGroupsResponse = $SOAP_Client -> ReadServiceGroups($serviceGroupsRequest);
}
catch (SoapFault $fault) {
    echo('Error in ReadServiceGroups: '. $fault -> __toString() .'<br />');
    exit;
}

echo "<h1>Read Services Test Execution</h1>";
echo "<br/>";
echo "<br/>";

//
// 2. For each service group: read and display service codes and label layouts
foreach (getElements($serviceGroupsResponse->ServiceGroup) as $group) {
    echo "<b>Available service codes and layouts for Service-Group '".$group-
>Description."'</b>";
    echo "<br/>";
    echo "<br/>";

    //
    // 2.1. Basic services for service group
    $basicServicesRequest = array('Language' => 'de', 'ServiceGroupID' => $group-
>ServiceGroupID);
    try {
        $basicServicesResponse = $SOAP_Client -> ReadBasicServices($basicServicesRequest);
    }
    catch (SoapFault $fault) {
        echo('Error in ReadBasicServices: '. $fault -> __toString() .'<br />');
        exit;
    }
```

```
    echo "Basic Services:";
    echo "<ul>";
    foreach (getElements($basicServicesResponse->BasicService) as $basicService) {
        echo "<li>";
        $basicServiceCodes = getElements($basicService->PRZL);
        echo toCommaSeparatedString($basicServiceCodes);

        //
        // 2.1.1. Additional services for basic service
        $additionalServicesRequest = array('Language' => 'de', 'PRZL' =>
$basicServiceCodes);
        try {
            $additionalServicesResponse = $SOAP_Client -> ReadAdditionalServices
($additionalServicesRequest);
        }
        catch (SoapFault $fault) {
            echo('Error in ReadAdditonalServices: '. $fault -> __toString() .'<br />');
            exit;
        }
        echo "<ul>";
        echo "<li>Additional service codes: ";
        $delimiter = "";
        foreach (getElements($additionalServicesResponse->AdditionalService) as
$additionalService) {
            echo $delimiter.$additionalService->PRZL;
            $delimiter = ",";
        }
        echo "</li>";
        echo "</ul>";
        // 2.1.1.
        //

        //
        // 2.1.2. Delivery instructions for basic service
        $deliveryInstructionsRequest = array('Language' => 'de', 'PRZL' =>
$basicServiceCodes);
        try {
            $deliveryInstructionsResponse = $SOAP_Client -> ReadDeliveryInstructions
($deliveryInstructionsRequest);
        }
        catch (SoapFault $fault) {
            echo('Error in ReadDeliveryInstructions: '. $fault -> __toString() .'<br />');
            exit;
        }

        echo "<ul>";
        echo "<li>Delivery instruction codes: ";
        $delimiter = "";
        foreach (getElements($deliveryInstructionsResponse->DeliveryInstructions) as
$deliveryInstruction) {
            echo $delimiter.$deliveryInstruction->PRZL;
            $delimiter = ",";
        }
```

```
        echo "</li>";
        echo "</ul>";
        // -- 2.1.2.
        //

        //
        // 2.1.3. Label layouts for basic service
        $labelLayoutsRequest = array('Language' => 'de', 'PRZL' => $basicServiceCodes);
        try {
            $labelLayoutsResponse = $SOAP_Client -> ReadLabelLayouts
($labelLayoutsRequest);
        }
        catch (SoapFault $fault) {
            echo('Error in ReadLabelLayouts: '. $fault -> __toString() .'<br />');
            exit;
        }

        echo "<ul>";
        echo "<li>Label Layouts: ";
        // elements
        echo "<ul>";
        foreach (getElements($labelLayoutsResponse->LabelLayout) as $labelLayout) {
            echo "<li>";
            echo $labelLayout->LabelLayout.": ";
            echo "max. ".$labelLayout->MaxServices." services ";
            echo "and ".$labelLayout->MaxDeliveryInstructions." delivery instructions, ";
            if ($labelLayout->FreeTextAllowed) {
                echo "freetext allowed";
            }
            else {
                echo "freetext not allowed";
            }
            echo "</li>";
        }
        echo "</ul>";
        // -- elements
        echo "</li>";
        echo "</ul>";
        // -- 2.1.3.
        //

        echo "</li>";
    }
    echo "</ul>";

}

echo "<br/>";
echo "<br/>";
echo "<br/>";
echo "<br/>";
echo "<h1>Allowed Services By Franking License XXXXX</h1>";
//
```

```
// 3. Read allowed services by franking license
$allowedServicesRequest = array('FrankingLicense' => 'XXXXX', 'Language' => 'de');
try {
    $allowedServicesResponse = $SOAP_Client -> ReadAllowedServicesByFrankingLicense
($allowedServicesRequest);
}
catch (SoapFault $fault) {
    echo('Error in ReadAllowedServicesByFrankingLicense: '. $fault -> __toString() .'<br
/>');
    exit;
}

echo "<ul>";
foreach (getElements($allowedServicesResponse->ServiceGroups) as $serviceGroup) {
    echo "<li>";
    echo "ServiceGroup: ".$serviceGroup->ServiceGroup->ServiceGroupID.", ".$serviceGroup-
>ServiceGroup->Description."";
    echo "<ul>";
    foreach (getElements($serviceGroup->BasicService) as $basicService) {
        echo "<li>";
        $przls = count($basicService->PRZL);
        if ($przls > 1) {
            echo "BasicService: ".$basicService->Description.": ".toCommaSeparatedString
($basicService->PRZL)."";
        } else {
            echo "BasicService: ".$basicService->Description.": ".$basicService->PRZL."";
        }
        echo "</li>";
    }
    echo "</ul>";
    echo "</li>";
    echo "<br/>";
}
echo "</ul>";

echo "</body></html>";
```

This example is available in the **wsbc-read.php** file.

4 Validation operation

The web service makes an operation available in order to validate a combination of service codes. The system checks that the service codes contained in the request (basic service, additional services and delivery instructions) can be combined. The system also checks whether the codes used are permitted in the chosen label format (A5, A6 or A7). Only a certain number of service codes and delivery instructions are permitted, depending on the format.

The following example demonstrates a check on a combination of basic service codes, additional service codes and delivery instruction codes.

```
include("wsbc-init.php");
include("wsbc-utils.php");

echo "<html><header></header><body>";

// 1. Define Validation Request
// (see documentation of structure in "Barcode web service manual")
$validationRequest = array(
    'Language' => 'de',
    'Envelope' => array(
    'LabelDefinition' => array(
        'LabelLayout' => 'A5'
        ),
    'Data' => array(
        'Provider' => array(
            'Sending' => array(
                'Item' => array(
                    array( // 1.Item ...
                        'ItemID' => '1',
                        'Attributes' => array(
                        'PRZL' => array(
                        // At least one code is required (schema validation)
                        // Basic service code(s) (optional, default="ECO"):
                        'PRI',
                        // Additional service codes (optional)
                        'FRA',
                        'SI',
                        // Delivery instruction codes (optional)
                        'ZAW3213',
                        'ZAW3215',
                    )
                )
            ),
                // Add additional items here for multiple requests in one web service
call ...
                // array( // 2.Item ...
                //     ... // same structure as above
                // ),
))))));

// 2. Web service call
try {
    $response = $SOAP_Client -> ValidateCombination($validationRequest);
}
```

```
catch (SoapFault $fault) {
    echo('Error in ValidateCombination: '. $fault -> __toString() .'<br />');
}

// 3. Process response
// (see documentation of structure in "Barcode web service manual")
foreach (getElements($response->Envelope->Data->Provider->Sending->Item) as $item) {
    if ($item->Errors != null) {
        // Errors in validation ...
        // (error messages are returned in the requested language)
        $errorMessages = "";
        foreach (getElements($item->Errors->Error) as $error) {
            $errorMessages .= $error->Message.',';
        }
        echo '<p>Validation-ERROR for item with itemID='.$item->ItemID.":
".$errorMessages.'.<br/></p>';
    }
    else {
        // Successful validation
        echo '<p>Validation was successfull for service code combination in item with
itemID='.$item->ItemID.'.<br/></p>';

        // Also display warnings
        if ($item->Warnings != null) {
            $warningMessages = "";
            foreach (getElements($item->Warnings->Warning) as $warning) {
                $warningMessages .= $warning->Message.",";
            }
            echo 'with WARNINGS: '.$warningMessages.'.<br/>';
        }
    }
}

echo "</body></html>";
```

This example validates without errors.

If however, for example, "ZAW3215" is replaced with "ZAW3222" the web service request replies with a corresponding validation error message.

This example is available in the **wsbc-validate.php** file.

5 Label generation

The following example demonstrates how an individual address label can be generated in php and the response read. The address label generated is saved in an image file.

```
include("wsbc-init.php");
include("wsbc-utils.php");

// Franking License Configuration
// TODO: you have to set this to a valid franking license for your barcode web service
user account!
$frankinglicense = '1234567890';

//$imgfile = 'default_logo.gif';
//$logo_binary_data = fread(fopen($imgfile, "r"), filesize($imgfile));

// 1. Define Label Request
// (see documentation of structure in "Barcode web service manual")
$generateLabelRequest = array(
    'Language' => 'de',
    'Envelope' => array(
        'LabelDefinition' => array(
            'LabelLayout' => 'A5',
            'PrintAddresses' => 'RecipientAndCustomer',
            'ImageFileType' => 'GIF',
            'ImageResolution' => 300,
            'PrintPreview' => false
        ),
        'FileInfos' => array(
            'FrankingLicense' => $frankinglicense,
            'PpFranking' => false,
            'Customer' => array(
                'Name1' => 'Meier AG',
                // 'Name2' => 'Generalagentur',
                'Street' => 'Viktoriaplatz 10',
                // 'POBox' => 'Postfach 600',
                'ZIP' => '8050',
                'City' => 'Zürich',
                // 'Country' => 'CH',
                // 'Logo' => $logo_binary_data,
                // 'LogoFormat' => 'GIF',
                'DomicilePostOffice' => '3000 Bern'
        ),
        'CustomerSystem' => 'PHP Client System'
    ),
    'Data' => array(
        'Provider' => array(
            'Sending' => array(
                'Item' => array(
                    array( // 1.Item ...
                        'ItemID' => '1234',
                        // 'ItemNumber' => '12345678',
                        // 'IdentCode' => '1234',
                        'Recipient' => array(
                            //'PostIdent' => 'IdentCodeUser',
```

```
                        'Title' => 'Frau',
                        'Vorname' => 'Melanie',
                        'Name1' => 'Steiner',
                        //'Name2' => 'Müller AG',
                        'Street' => 'Viktoriastrasse',
                        'HouseNo' => '21',
                        //'FloorNo' => '1',
                        //'MailboxNo' => '1111',
                        'ZIP' => '3030',
                        'City' => 'Bern 1',
                        //'Country' => 'CH',
                        'Phone' => '0313381111', // für ZAW3213
                        //'Mobile' => '0793381111',
                        'EMail' => 'h.muster@post.ch'
                       ///'LabelAddress' => array(
                        //  'LabelLine' => array('LabelLine 1',
                        //            'LabelLine 2',
                        //            'LabelLine 3',
                        //            'LabelLine 4',
                        //            'LabelLine 5'
                        //            )
                        //)
                    ),
                    //'AdditionalINFOS' => array(
                    // Cash-On-Cary amount in CHF for service 'BLN':
                    //    'AdditionalData' => array(
                    //    'Type' => 'NN_BETRAG',
                    //        'Value' => '12.5'
                    //    ),
                    // Cash-On-Cary ESR-Reference-No. for service 'BLN':
                    //    AdditionalData' => array(
                    //        'Type' => 'NN_ESR_REF_REFNR',
                    //        'Value' => 'xxxxxxxxxxxxxxxxxxxxxx' // a valid ESR
Ref. Nr. (with or without spaces)
                    //    ),
                    //),
                    'Attributes' => array(
                    'PRZL' => array(
                         // At least one code is required (schema validation)

                         // Basic service code(s) (optional, default="ECO"):

                         'PRI',
                         // Additional service codes (optional)
                         'N',
                         'FRA',
                         // Delivery instruction codes (optional)
                         'ZAW3211',
                         'ZAW3213'
                      ),
```

```
                        // Cash on delivery amount in CHF for service 'N':
                        'Amount' => 12.5,
                        'FreeText' => 'Freitext',
                        // 'DeliveryDate' => '2010-06-19',
                        // 'ParcelNo' => 2,
                        // 'ParcelTotal' => 5,
                        // 'DeliveryPlace' => 'Vor der Haustüre',
                        'ProClima' => true,
                    )
                        // Setting notifications
                        //'Notification' => array(
                        //     'Service' => 64,
                        //     'Communication' => array(
                        //         'Email' => 'test@test.ch'
                        //     ),
                        //     'FreeText1' => 'Free-Text 1',
                        //     'FreeText2' => 'Free-Text 2',
                        //     'Language' => 'DE'
                        //)
                    )
                //,
                // Add addtional items here for multiple requests in one web service
call ...
                // array( // 2.Item ...
                //     ... // same structure as above
                // ),
                )
            )
        )
    )
  )
);

// Setting attribute of notification
// $notification = $generateLabelRequest -> Data -> Provider -> Sending -> Item[0]->
Notification;
// $notification['Type'] = ,EMAIL';

// 2. Web service call
$response = null;
try {
    $response = $SOAP_Client -> GenerateLabel($generateLabelRequest);
}
catch (SoapFault $fault) {
    echo('Error in GenerateLabel: '. $fault -> __toString() .'<br />');
}
// 3. Process requests: save label images and check for errors
// (see documentation of structure in "Barcode web service manual")
foreach (getElements($response->Envelope->Data->Provider->Sending->Item) as $item) {
    if ($item->Errors != null) {

        // Error in Label Request Item:
        // This barcode label was not generated due to errors.
        // The received error messages are returned in the specified language of the
request.
```

```
        // This means, that the label was not generated,
        // but other labels from other request items in same call
        // might have been generated successfully anyway.
        $errorMessages = "";
        $delimiter = "";
        foreach (getElements($item->Errors->Error) as $error) {
            $errorMessages .= $delimiter.$error->Message;
            $delimiter = ",";
        }
        echo '<p>ERROR for item with itemID='.$item->ItemID.":
".$errorMessages.'.<br/></p>';

    }
    else {
        // Get successfully generated label as binary data:
        $identCode = $item->IdentCode;
        $labelBinaryData = $item->Label;

        // Save the binary image data to image file:
        $filename =
'outputfolder\testOutput_GenerateLabel_'.$identCode.'.gif';
        file_put_contents($filename, $labelBinaryData);

        // Printout some label information (and warnings, if any):
        echo '<p>Label generated successfully for identCode='.$identCode.': <br/>';
        if ($item->Warnings != null) {
            $warningMessages = "";
            foreach (getElements($item->Warnings->Warning) as $warning) {
                $warningMessages .= $warning->Message.",";
            }
            echo 'with WARNINGS: '.$warningMessages.'.<br/>';
        }
        echo $filename.':<br/><img src="'.$filename.'"/><br/>';
        echo '</p>';
    }
}

echo "</body></html>";
```

Important Note on the Operation GenerateLabel:

- In order for the above example to work, a valid franking licence must always be set for the Login web service being used.
- Should an item in a response contain one or more errors, it means that this address label could not be generated. However, it may be that other address labels in the same request were correctly generated.
- Even if an address label was successfully generated, it may that the item contains warnings. This is most likely if, for instance, parameters that are not required for the chosen services were entered.

Various other fields may be placed in a GenerateLabelRequest. Please refer to the complete documentation on the "Barcode" web service [1], in particular section 4.3.1.

The sample file **wsbc-generate.php** also contains the above example with additional comments on all the optional fields that can be additionally set.

6 Single Barcodes Generation

The following example demonstrates how several barcode elements of the function GenerateSingleBarcodes can be generated in PHP and the response read out. The generated barcode elements are saved as image files.

```
include("wsbc-init.php");
include("wsbc-utils.php");

// Franking License Configuration
// TODO: you have to set this to a valid franking license for your barcode web service user account!
$frankinglicense = '1234567890';

// 1. Define Label Request
// (see documentation of structure in "Barcode web service manual")
$generateSingleBarcodesRequest = array(
    'Language' => 'de',
    'Envelope' => array(
        'BarcodeDefinition' => array(
            'ImageFileType' => 'PNG',
            'ImageResolution' => 300,
        ),
        'FileInfos' => array(
            'FrankingLicense' => $frankinglicense,
            'PpFranking' => false,
            'Customer' => array(
                'Name1' => 'Meier AG',
                // 'Name2' => 'Generalagentur',
                'Street' => 'Viktoriaplatz 10',
                // 'POBox' => 'Postfach 600',
                'ZIP' => '8050',
                'City' => 'Zürich',
                // 'Country' => 'CH',
                'DomicilePostOffice' => '3000 Bern'
        ),
    ),
    'Data' => array(
        'Provider' => array(
            'Sending' => array(
                'Item' => array(
                    array( // 1.Item ...
                        'ItemID' => '1234',
                        // 'ItemNumber' => '12345678',
                        // 'IdentCode' => '1234',
                        'Recipient' => array(
                            //'PostIdent' => 'IdentCodeUser',
                            'Title' => 'Frau',
                            'Vorname' => 'Melanie',
                            'Name1' => 'Steiner',
                            //'Name2' => 'Müller AG',
                            'Street' => 'Viktoriastrasse',
                            'HouseNo' => '21',
                            //'FloorNo' => '1',
                            //'MailboxNo' => '1111',
```

```
                        'ZIP' => '3030',
                        'City' => 'Bern 1',
                        //'Country' => 'CH',
                        //'Phone' => '0313381111', // für ZAW3213
                        //'Mobile' => '0793381111',
                        'EMail' => 'h.muster@post.ch'
                        ///'LabelAddress' => array(
                        //  'LabelLine' => array('LabelLine 1',
                        //            'LabelLine 2',
                        //            'LabelLine 3',
                        //            'LabelLine 4',
                        //            'LabelLine 5'
                        //            )
                        //)
                    ),
                    //'AdditionalINFOS' => array(
                    // Cash-On-Cary amount in CHF for service 'BLN':
                    //    'AdditionalData' => array(
                    //    'Type' => 'NN_BETRAG',
                    //        'Value' => '12.5'
                    //    ),
                    // Cash-On-Cary ESR-Reference-No. for service 'BLN':
                    //    AdditionalData' => array(
                    //        'Type' => 'NN_ESR_REF_REFNR',
                    //        'Value' => 'xxxxxxxxxxxxxxxxxxxxxx' // a valid ESR
Ref. Nr. (with or without spaces)
                    //    ),
                    //),
                    'Attributes' => array(
                      'PRZL' => array(
                         // At least one code is required (schema validation)
                         'eAR',
                         'RINL',
                         'ZAW2511'
                      ),

                      // Cash on delivery amount in CHF for service 'N':
                      // 'Amount' => 12.5,
                      'FreeText' => 'Freitext',
                      // 'DeliveryDate' => '2010-06-19',
                      // 'ParcelNo' => 2,
                      // 'ParcelTotal' => 5,
                      // 'DeliveryPlace' => 'Vor der Haustüre',
                      'ProClima' => true
                    )
                    //'Notification' => array(
                    // // Notification structure ...
                    //)
                )
            //,
            // Add additional items here for multiple requests in one web service
call ...
            // array( // 2.Item ...
            //     ... // same structure as above
            // ),
            )
```

```
            )
        )
    )
  )
);

// 2. Web service call
$response = null;
try {
    $response = $SOAP_Client -> GenerateSingleBarcodes($generateSingleBarcodesRequest);
}
catch (SoapFault $fault) {
    echo('Error in GenerateSingleBarcodes: '. $fault -> __toString() .'<br />');
}

// 3. Process requests: save label images and check for errors
// (see documentation of structure in "Barcode web service manual")
foreach (getElements($response->Envelope->Data->Provider->Sending->Item) as $item) {
    if ($item->Errors != null) {

        // Error in Single Request Item:
        // This barcode label was not generated due to errors.
        // The received error messages are returned in the specified language of the
request.
        // This means, that the label was not generated,
        // but other labels from other request items in same call
        // might have been generated successfully anyway.
        $errorMessages = "";
        $delimiter = "";
        foreach (getElements($item->Errors->Error) as $error) {
            $errorMessages .= $delimiter.$error->Message;
            $delimiter = ",";
        }
        echo '<p>ERROR for item with itemID='.$item->ItemID.":
".$errorMessages.'.<br/></p>';

}
    else {
        // Get successfully generated label as binary data:
        $itemID = $item->IdentID;
        $identCode = $item->IdentCode;

        $counter = 1;
        $basePath =
'outputfolder\testOutput_GenerateSingleBarcodes_'.$identCode.'_';
        foreach (getElements($item->Barcodes->Barcode) as $barcode) {
            // Save the binary image data to image file:
            $filename = ''.$basePath.''.$counter.'.gif';
            file_put_contents($filename, $barcode);
            $counter++;
        }
        $numberOfItems = $counter - 1;
         // Printout some label information (and warnings, if any):
        echo '<p>'.$numberOfItems.'Barcodes successfully generated for
identCode='.$identCode.': <br/>';
```

```
        if ($item->Warnings != null) {
            $warningMessages = "";
            foreach (getElements($item->Warnings->Warning) as $warning) {
                $warningMessages .= $warning->Message.",";
            }
            echo 'with WARNINGS: '.$warningMessages.'.<br/>';
        }
        echo 'All files start with: <br/><img src="'.$basePath.'"/><br/>';
        echo '</p>';
    }
}

echo "</body></html>";
```

Important Information on the Operation GenerateLabel:

- in order for the example above to function properly, a valid franking licence for the applicable web service login must be set.
- Should an item in a GenerateSingleBarcodes response contain one or more errors, it means that barcode elements could not be generated.
- The response item may contain warnings even when the barcode elements were successfully generated. This is especially likely when, for example, parameters were consigned that were not required for the selected services.

In a GenerateSingleBarcodes request, various other fields can be set. We refer you to the full documentation Webservice Barcodes [1], specifically to Section 4.3.x.

Additionally you will find in the sample file **wsbc-generate-singlebarcodes.php** the above example with additional comments and all further optional fields that can be set.

7 Individual Barcodes Generation

The following example demonstrates how individual barcodes can be generated in PHP using the function GenerateBarcodes, and the response read out. The barcodes are saved as image files.

```
include("wsbc-init.php");
include("wsbc-utils.php");

echo "<html><header></header><body>";

// 1. Define Label Request
// (see documentation of structure in "Barcode web service manual")
$generateBarcodeRequest = array(
    'Language' => 'de',
    'BarcodeDefinition' => array(
        'BarcodeType' => 'LSO_2',
        'ImageFileType' => 'PNG',
        'ImageResolution' => 200
    )
);

// 2. Web service call
$response = null;
try {
    $response = $SOAP_Client -> GenerateBarcode($generateBarcodeRequest);
}
catch (SoapFault $fault) {
    echo('Error in GenerateBarcode: '. $fault -> __toString() .'<br />');
}

// 3. Process requests: save label images and check for errors
// (see documentation of structure in "Barcode web service manual")
foreach (getElements($response->Data) as $data) {
    if ($data->Errors != null) {

        // Error in Barcode Request Item:
        // This barcode label was not generated due to errors.
        // The received error messages are returned in the specified language of the
request.
        // This means, that the label was not generated,
        // but other labels from other request items in same call
        // might have been generated successfully anyway.
        $errorMessages = "";
        $delimiter = "";
        foreach (getElements($data->Errors->Error) as $error) {
            $errorMessages .= $delimiter.$error->Message;
            $delimiter = ",";
        }
        echo "<p>ERROR for request: ".$errorMessages.'.<br/></p>';

    }
```

```
    else {
        // Get successfully generated barcide as binary data:
        $barcodeBinaryData = $data->Barcode;
        $deliveryNoteRef = $data->DeliveryNoteRef;
        $barcodeDefinition = $data->BarcodeDefinition;
        $barcodeType = $barcodeDefinition->BarcodeType;
        $imageFileType = $barcodeDefinition->ImageFileType;
        $imageResolution = $barcodeDefinition->ImageResolution;

        // Save the binary image data to image file:
        $filename =
'outputfolder\testOutput_GenerateBarcode_'.$deliveryNoteRef.'.gif';
        file_put_contents($filename, $barcodeBinaryData);

        // Printout some label information (and warnings, if any):
        echo '<p>Label generated successfully for Delivery Note Reference =
'.$deliveryNoteRef.': <br/>';
        if ($data->Warnings != null) {
            $warningMessages = "";
            foreach (getElements($data->Warnings->Warning) as $warning) {
                $warningMessages .= $warning->Message.",";
            }
            echo 'with WARNINGS: '.$warningMessages.'.<br/>';
        }
        echo $filename.':<br/><img src="'.$filename.'"/><br/>';
        echo '</p>';
    }
}

echo "</body></html>";
```

Important Information on the Operation GenerateBarcode:

- Should a data object in a GenerateBarcode response contain one or more errors, it means that this barcode could not be generated.
- The response data object may contain warnings even when the barcode was successfully generated. This is especially likely when, for example, parameters were consigned that were not required for the selected services.

For further information, we refer you to the full documentation Webservice Barcodes [1], specifically to Section 4.3.x.

Additionally you will find the above example in the sample file **wsbc-generatebarcode.php**.

Post CH Ltd
Support Webservices
Viktoriastrasse 21
P.O. Box
3030 Berne, Switzerland

webservice@post.ch
www.swisspost.ch/webservice

SWISS POST

11.2013